Assignment 3

Harry Potter: The Object of Fire

In this assignment you will design and implement some new game functionality, and write a set of recommendations for changes to the game engine.

Project requirements

All the requirements stated in the Assignment 1 and 2 specifications still apply.

Apparate spell

Implement an *Apparate* spell, with the following properties

- cast (via a wand) on oneself.
- When cast, the Actor dissaparates (or dissapears) from the current location and apparates (re-appears) at another location on the next turn. (Note that while apparating, other HPEntities continue to move normally.)
- If the grid coordinate is not empty on arrival then the actor will appear at the first empty grid location in a randomly chosen direction. In this case however, the actor will sustain between (1 - 10) * 10 hit-points damage due to *splinching*.

Teacher

Young wizards don't know many spells. Some actors can be teachers. A teacher has the ability to teach any other actor it meets any of the spells that it knows.

- design and implement a way for some actors to be teachers
- when a teacher is in the same location as another actor on the same team, it will offer to teach the other actor a spell
  - if the other actor is the game player, it can choose a spell to learn from the list of spells the teacher knows, or can decline the offer, using the user interface to made the choice
  - if the other actor is a non-player actor, it chooses a spell it doesn't already know from the list of spells the teacher knows, or declines the offer, with uniform probability
- make the game player a teacher. The game player starts the game knowing no spells
- make Dumbledore a teacher. Dumbledore knows all the spells in the game

Secret tunnel

In the Harry Potter stories, there are secret tunnels. Create a secret tunnel between these locations $(7, 4)$ and $(0, 1)$ with the following properties:

- The tunnel is a grid of locations of dimensions 1×8. When the game player is in the tunnel, the tunnel grid is displayed in the user interface rather than the main 10×10 grid
- There is a "door" at each end of the tunnel that provides an action that allows an actor using that action to move to the first tunnel location at that end of the tunnel

Inventory

Add an Inventory capability that allows the actor to hold up to three (3) items. This could include a wand, a potion, or some other collectable artefact.

Broomstick

In possession of a Broomstick an actor can travel at a double speed. Broomsticks can be acquired from a

teacher after some training (for example after learning 2 spells). At any given time, an actor can possess one broomstick only.

Recommendations for change to the game engine

Over the course of your three assignments, you have had to become familiar with the game engine (located in the packages with prefix `edu.monash.fit2099`). You have almost certainly found aspects of the game engine design and implementation difficult to understand, or frustrating to use.

Write a short document (e.g. two or three A4 pages of text) describing changes you would recommend for the game engine. For each change you recommend, you must explain:

- what problem you perceive
- the design change you propose to address the problem
- the advantages, and any disadvantages, of the proposed change

If you use UML diagrams, don't count the space they take up in the "two or three A4 pages of text".

Note that any recommendations you make must be suitable for any application of the game engine. They must not be specific to the Harry Potter scenario (for example, last year the setting of the game was the Star Wars universe).

Design is important

One of the primary aims of this unit is for you to learn the fundamentals of object-oriented design. In order to get a high mark in this assignment, it will not be sufficient for your code to "work". It must also adhere to the design principles covered in lectures, and in the required readings on Moodle.

If you would like informal feedback on your design at any time, please consult your lab demonstrator in a lab or consultation session.

You must ensure that design documents exist for all requirements by the beginning of last week.

Updating your design

You might find that some aspects of your existing design need to be modified to support the new functionality. You might also think of a better approach to some of the requirements you have already implemented — your understanding of the requirements and codebase will have improved as you have progressed. You might also spot places where your existing code (and thus design) can benefit from refactoring.

If you want to update your existing design, you may do so; if you decide to do this, be sure to update your design documents so that they match the code, and write a brief explanation of your changes and the reasons behind them. his will help your marker understand the thinking behind your code.

Coding and commenting standards

You must adhere to the Java coding standards that were posted on Moodle earlier in the semester.

Write javadoc comments for *at least* all public methods and attributes in your classes.

You will be marked on your adherence to the standards, javadoc, and general commenting guidelines that were posted to Moodle earlier in the semester.

Bonus marks

As stated in the Assignment 2 specification, there are bonus marks available for groups that come up with ideas for new features for the game, and then design and implement them. These could be new kinds of actor or entity, with new behaviours. They could be new kinds of location. They must be different from any features that have been requirements in FIT2099 in previous semesters.

A feature must be quite complex to qualify for a bonus mark. You must discuss your plans for bonus mark features with your tutor and gain approval before beginning work on them. Bonus marks will be

awarded after the submission of Assignment 3.

A maximum of three bonus marks are available (these are whole marks for the unit). No one has to attempt bonus marks, and it is possible to get full marks for the unit without any bonus marks.

Submission instructions

The due date for this assignment is *Tuesday 5th February at 23:59*. We will mark your Assignment 3 on the state of the "master" branch of your Monash GitLab repository at that time. If you have done any work in other branches, make sure you merge it into master before the due time.

Important: You must ensure that design documents exist for *all* requirements prior to their implementation. Designs *must* be completed by the due date above - any further updates will be communicated via email.

Do not create a new copy of your work for Assignment 3. Continue working on the same files, in the same directory structure (you might like to add a tag to the your final Assignment 2 commit before starting on Assignment 3, so you can find that version easily).[1]

As we said above, you may update your design and implementation if you find that your Assignment 2 solution is not suitable for extension, or if you think of a better approach during Assignment 3.

You must update your Work Breakdown Agreement to include the work necessary for the Assignment 3 requirements, and any bonus mark work you attempt.

Unless a team member has applied for and received special consideration according to the Monash Special Consideration Policy,[2] late submissions will be penalized at 10% per day late.

It is both team members' responsibility to ensure that the correct versions of the documentation and code are present in the repository by the due date and time. Once both teammates have agreed on a final Assignment 3 submission, do not make further commits to the master branch of the repository until the due date has passed, without the agreement of your teammate. If you want to continue to make changes to the repository for some reason, make another branch.

We will take your Work Breakdown Agreement into account when marking if there seems to be a major discrepancy in the quality of different parts of the submission, or if the code is missing major sections. Students whose work is inadequate in either quality or quantity will be penalized, and their partners will be compensated. If you choose to reallocate tasks, make sure you keep your WBA up to date.

Marking

Marking Criteria

This assignment will be marked on:

- Functional completeness
- Design quality
  - adherence to design principles discussed in lectures
  - UML notation consistency and appropriateness
  - consistency between design artefacts
  - clarity of writing.
  - level of detail (this should be sufficient but not overwhelming)
- Code quality

[1] https://git-scm.com/book/en/v2/Git-Basics-Tagging
[2] http://www.monash.edu/exams/changes/special-consideration

  - readability
  - adherence to Java coding standards
  - quality of comments
  - maintainability (application of the principles discussed in lectures)
- Correct use of GitLab
- Quality of recommendations for changes to the codebase
  - understanding of codebase displayed
  - usefulness of recommendations
  - understandability of recommendations
  - generality of recommendations (i.e. must not be specific to Harry Potter scenario)
  - practicality of recommendations
  - clarity of writing

Marks may also be **deducted** for:

- late submission
- inadequate individual contribution to the project
- academic integrity breaches